Die Gentoo-Paketverwaltung

Portage

André Kloth

Hasso-Plattner Institut
Universität Potsdam

24. Juni 2007

Themen

1. Paketverwaltungssysteme
2. Portage
3. Bedienung von Portage
4. Ebuilds

Teil I

Paketverwaltungssysteme

Aufgaben von Paketverwaltungssystemen

- installieren*
- aktualisieren*
- konfigurieren(*)
- deinstallieren*

* **Abhängigkeiten auflösen**

Komponenten

- Front-End-Anwendung
- Verzeichnis installierter Pakete mit Metainformationen
- Verzeichnis verfügbarer Pakete (meist mit zusätzlicher Infrastruktur zum Download)

Updatemechanismen

Bestehen meistens aus einem zweistufigem Verfahren

1. Update der lokalen Paketinformationen
2. Update der installierten Pakete

Abhängigkeiten zwischen Paketen

Verwalten von Abhängigkeiten ist eine der wesentlichen Aufgaben eines Paketverwaltungssystems.

- Paket benötigt ein anderes Paket/Feature/etc. (*requires*)
- Paket bietet eine Funktion an (*provides*)
- Paket steht im Konflikt mit einem anderen Paket/Feature/etc. (*conflicts*)

(fast) jede Distribution hat ein (bevorzugtes) Paketsystem

- RPM – Red Hat, SuSE, Fedora, Mandriva
- APT – Debian, Ubuntu
- Ports – FreeBSD, OpenBSD
- Portage – Gentoo
- (MSI – Windows)

Was macht Portage/Gentoo so besonders?

- Pakete werden aus Quellcode kompiliert
- Performancegewinn durch Optimierung beim Kompilieren
- SLOTs zur gleichzeitigen Installation unterschiedlicher Paketversionen
- unterstützt Alpha, AMD64, Itanium, MIPS, PowerPC, SPARC, UltraSparc, x86, etc.
- nicht-einschränkende Lizenzpolitik
- *stable* und *unstable* Pakete
- große Community; Gentoo-Foren

Teil II

Portage

Komponenten (Auszug)

- Portage Paket-Verzeichnis /usr/portage geordnet nach Kategorien

- ebuild, emerge
- dispatch-conf
- revdep-rebuild
- quickpkg

- installierte Pakete: /var/db/pkg

Konfiguration

zentrale Konfigurationsdatei /etc/make.conf

- `CFLAGS/CXXFLAGS` – z.B. `-O2 -mtune=i686 -pipe`
- `CHOST` – z.B. `i686-pc-linux-gnu`
- `MAKEOPTS` – z.B. `-j3`
- `ACCEPT_KEYWORDS` (stable/unstable) – z.B. `x86`
- `GENTOO_MIRRORS`
- `USE`

USE-Flags

Information für Portage, welche Features gewünscht/unerwünscht sind.

- globale Definition in /etc/make.conf
  - `USE='X mmx -doc ...'`
  - falls Features unterstützt werden, sollen sie einkompiliert werden
  - zusätzliche Abhängigkeiten werden aufgelöst/installiert
- Paket-abhängige Definition in /etc/portage/package.use
- explizites Überschreiben von globalen USE-Flags
  - `echo 'app-editors/vim -X' >> ↩ /etc/portage/package.use`

von Slots, Virtuals, Architekturen, ...

- Slots für parallel installiertbare Paketversionen (z.B. gcc, Kernel)
- Virtuals bezeichnen immer eine Gruppe von Paketen, die eine gleiche oder ähnliche Funktionalität bieten
  - `virtual/cron`, `virtual/logger`, ...
  - wird als Abhängigkeit in verschiedenen Paketen referenziert
- Architekturen durch `KEYWORDS` beschrieben
  - Paket-abhängige Definition in /etc/portage/package.keywords
  - um *instabile*/nicht unterstützte Pakete zu installieren

Teil III

Alltäglicher Gebrauch

emerge

offizielles Konsolen-Front-End für Portage

- Bestandteil des Pakets sys-apps/portage
- in Python programmiert (auch die meisten Module)
- implementiert alle notwendigen Funktionen zur Paketverwaltung

Suchen

`emerge --search mplayer`

- durchsucht /usr/portage
- langsame dateibasierte Suche

Alternative: `app-portage/esearch`

- `eupdatedb` indiziert Paketnamen
- schnell Suche in Hash-Tabelle (ähnlich zu locate/updatedb)

Installieren/Deinstallieren

`emerge <paket>`

- `emerge --ask --verbose --tree <paket>`

`emerge -C <paket>`

- `emerge --ask -C <paket>`

Updatemechanismus

Update der lokalen Portage-Paketdatenbank

- `emerge --sync` – rsync-Protokoll
- `emerge-websync` – Download des Updates über HTTP

Update der installierten Pakete

- `emerge --update --ask --verbose --tree world`

Inkonsistenzen durch Updates (1)

Update kann z.B. geänderte Syntax von Konfigurationsdateien beinhalten.

`dispatch-conf`:

- zu aktualisierende Konfigurationsdateien finden
- Benutzer entscheidet anhand eines *Diff*s, ob die Änderung übernommen, automatisch (merge) oder manuell eingearbeitet wird
- Backup erstellen

Inkonsistenzen durch Updates (2)

Update von z.B. Bibliotheken kann darauf aufbauende Programme funktionsunfähig machen

`revdep-rebuild`:

- Linking prüfen
- zugehörige Pakete finden
- Code neu linken (kompilieren)

distcc aka Features (1)

distcc verteilt automatisch Quellcode zum Kompilieren

- Geschwindigkeit
- benötigt C/C++ Kompiler für die Zielplattform
- verschickt Quellcode
- keine Bibliotheken/Header auf entfernten Hosts notwendig

ccache aka Features (2)

Compiler Cache

- cached vergangene Kompiliervorgänge
- kompiliert bis zu 5 - 10 mal schneller
- maximaler Speicherverbrauch konfigurierbar

Binärpakete aka Features (3)

`quickpkg`

- erstellt Binärpakete aus installierten Paketen
- interessant für homogene Gentoo-Installationen
- ermöglicht *Sicherungskopie* wichtiger Pakete (z.B. glibc)

Teil IV

Ebuilds

Python und Bash

Portage – Sammlung aus Python-Programmen/-Klassen

- Funktionen um
  - Kompilier- und Installationsvorgang zu koordinieren
  - Downloads z.B. aus CVS-Repositories vorzunehmen
  - (erweiterte) Abhängigkeiten zu prüfen/aufzulösen (z.B. Kernel-Config)
  - ...

Ebuilds – Bash-Skripte

Eigene Pakete schreiben (1)

Definition von `PORTDIR_OVERLAY` in /etc/make.conf

Ebuilds beschreiben Meta-Informationen über ein Paket

- Beschreibung, Homepage, Download-URL, Lizenz, Architektur
- Abhängigkeiten, USE-Flags, ...

Ebuilds enthalten Callback-Funktionen für verschiedene Stufen des Installationsprozesses

- `src_unpack`
- `src_compile`
- `src_install`
- `pkg_postinstall`
- ...

Eigene Pakete schreiben (1)

Definition von `PORTDIR_OVERLAY` in /etc/make.conf

Ebuilds beschreiben Meta-Informationen über ein Paket

- Beschreibung, Homepage, Download-URL, Lizenz, Architektur
- Abhängigkeiten, USE-Flags, ...

Ebuilds enthalten Callback-Funktionen für verschiedene Stufen des Installationsprozesses

- `src_unpack`
- `src_compile`
- `src_install`
- `pkg_postinstall`
- `...`

Eigene Pakete schreiben (2)

Testen mit Portage-Backend – `ebuild`

- `ebuild <paket> digest`
- `ebuild <paket> compile`
- `ebuild <paket> install`
- `ebuild <paket> qmerge`

Eigene Pakete schreiben (3)

weitere Schritte

- mit `emerge` testen
- verschiedene USE-Flags testen
- als Bug an `http://bugs.gentoo.org` schicken

- wenn sich ein Betreuer unter den Gentoo-Entwicklern findet, wird das Paket aufgenommen
- Statistik:
  - 149 Kategorien
  - 11691 Package
  - 23825 Ebuilds

Eigene Pakete schreiben (3)

weitere Schritte

- mit `emerge` testen
- verschiedene USE-Flags testen
- als Bug an `http://bugs.gentoo.org` schicken

- wenn sich ein Betreuer unter den Gentoo-Entwicklern findet, wird das Paket aufgenommen
- Statistik:
  - 149 Kategorien
  - 11691 Package
  - 23825 Ebuilds

FIN

- Danke für die Aufmerksamkeit!

Literaturverzeichnis

Paketverwaltung
OpenSuSE Webseite
`http://de.opensuse.org/Paketverwaltung`, 18.06.2007

Übersicht über FreeBSD Ports
About FreeBSD Port
`http://www.freebsd.org/ports/`, 18.06.2007

FreeBSD Handbuch
FreeBSD Handbook, *Installing Applications: Packages and Ports*
`http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/`, 18.06.2007

Literaturverzeichnis


Gentoo Handbook
Gentoo Handbook Website
http://www.gentoo.org/doc/en/handbook/, 19.06.2007


Universität Potsdam Linux User Group (upLUG)
BlinkenKUZE – Entwicklungs-/Dokumentationsseite
http://wiki.uplug.de/index.php?page=BlinkenKUZE/Software, 21.06.2007


Debian Documentation
Debian Documentation / Installation Guide
http://www.debian.org/doc/,
http://www.debian.org/releases/stable/i386/,
18.06.2007